Preprint
UCRL-CONF-200095

A Comparison of Dimensionality Reduction Methods for Retrieval of Similar Objects in Simulation Data

Erick Cantú-Paz
Sen-ching S. Cheung
Chandrika Kamath

This article was submitted to 2004 SIAM Data Mining Conference

September 23, 2003

This research was supported under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

A Comparison of Dimensionality Reduction Methods for Retrieval of Similar Objects in Simulation Data

Erick Cantú-Paz[*] Sen-ching S. Cheung Chandrika Kamath

Abstract

High-resolution computer simulations produce large volumes of data. As a first step in the analysis of these data, supervised machine learning techniques can be used to retrieve objects similar to a query that the user finds interesting. These objects may be characterized by a large number of features, some of which may be redundant or irrelevant to the similarity retrieval problem. This paper presents a comparison of six dimensionality reduction algorithms on data from a fluid mixing simulation. The objective is to identify methods that efficiently find feature subsets that result in high accuracy rates. Our experimental results with single- and multi-resolution data suggest that standard forward feature selection produces the smallest feature subsets in the shortest time.

1 Introduction

Computer simulations provide qualitative and quantitative insights into physical phenomena. Simulations are particularly useful when the phenomena are too complex to be studied analytically or too expensive, impratical, or dangerous to study experimentally. However, computer simulations generate large volumes of data that require specialized tools to be analyzed. This paper describes a component of a system that can be used as a first step in the analysis of simulation data by retrieving objects similar to a query that the user finds interesting. The system uses classification algorithms to incorporate the user feedback on the quality of the solutions returned.

The objects are described by a feature vector that may contain a large number of features, some of which may be redundant or irrelevant to the task of finding objects similar to a query. Avoiding irrelevant or redundant features is important because they may have a negative effect on the accuracy of the classifier. In addition, using fewer features will result in faster retrievals from a large database of feature vectors. Searching exhaustively for the best feature subset is impractical for high-dimensional data (if objects are described by d features, there are 2^d possible feature subsets) and therefore heuristic methods are used. An alternative for reducing the dimensionality is to project the data to a lower-dimensionality space that is constructed with characteristics that seem desirable, such as orthogonality.

This paper compares six dimensionality reduction algorithms on data from a fluid mixing simulation. We use four "wrapper" algorithms, which use the performance of a classifier on candidate feature subsets to guide the search, as well as principal components analysis (PCA) and a variable elimination algorithm based on PCA. The objective of this work is to identify an algorithm that efficiently finds a compact representation of the data that results in high accuracy rates.

The next section briefly describes the similarity-based object retrieval system that we implemented. Section 3 describes the data and the algorithms used in this study. Section 4 presents the experimental results with the various dimensionality reduction algorithms. Finally, section 5 summarizes the paper and presents our conclusions.

2 Similarity-based Object Retrieval

Similarity-based object retrieval (SBOR) is a system being developed as part of the Sapphire project at Lawrence Livermore National Laboratory.[1] The motivation for the system is to support searches in simulation data that are similar to those in content-based image retrieval (CBIR) systems. However, instead of focusing on the entire image as is often done in CBIR, we are interested in retrieving sub-images, or objects, in the data. This is because in the analysis of simulation data the user is not interested in the entire output at any time step, but in a smaller region of this output. In addition, the modules of such a retrieval tool can be applied to other problems in the analysis of simulation data. As an example, consider the code validation problem in Figure 1, where we need to determine how close a simulation is to an experiment. One approach to this would be to consider each of the mushroom-shaped objects in the simulation and measure its similarity to the corresponding object in the experiment. This can be done using the modules of an SBOR system. Having quantitatively measured the similarity of these objects

[*]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550

[1]`http://www.llnl.gov/casc/sapphire`

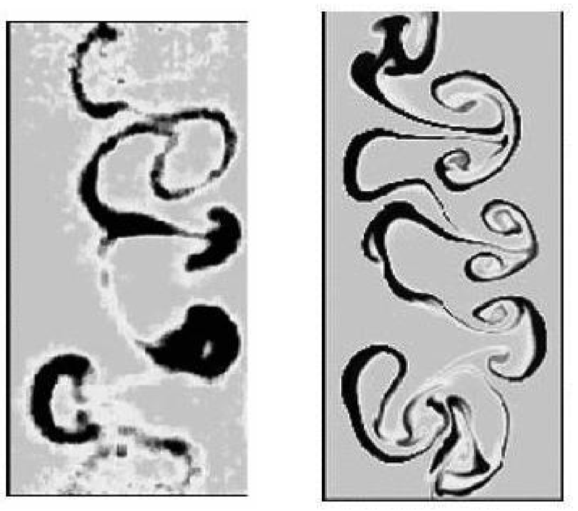

Figure 1: The left image shows the flow pattern of a Richtmyer-Meshkov experiment performed at Los Alamos National Laboratory. The same experiment is simulated by high resolution numerical methods and the result is shown in the right image [9].

taken in isolation, we could then combine these measures with additional information such as the number and locations of the mushrooms to quantitatively compare the experimental image with the ones from the simulation.

In a typical similarity search conducted using the SBOR system, a user first identifies an object of interest in an image from the image database and defines a rectangular region on the image as the query. Then, the user specifies the images to query on as well as the features to be used in the similarity search. The user can select from a large array of features, which are described in more detail in the next section. The SBOR system uses this information to populate a feature database with feature vectors extracted from the images. For this task, we adopt a simple sliding-window approach. A tile window, with the same dimensions as the query image, is moved across each image in a fixed-size step. A feature vector is computed for the part of the image under the tile window at each location. In the experiments reported here, a small step size of eight pixels is used for both the horizontal and vertical directions.

In order to retrieve objects that are similar to the query object, but much smaller or much larger, we also support multiresolution tiling. Using a simple Gaussian pyramid, obtained through convolution with a Gaussian filter and sub-sampling, we obtain an image that is half as large as the original image in each dimension. When this coarse resolution image is tiled using tiles of the same size as the original query tile, we are in effect using larger tiles on the original image. This allows us to find objects larger than the query object, while performing fewer computations with the smaller image as compared to increasing the tile size. In addition, as these larger tiles often contain more than one object, we do object segmentation on the image prior to the multiresolution. The features are then calculated for each object in the larger tile, assuming that the object was the only one in the tile. Instead of a Gaussian filter, we can also use a wavelet pyramid. In addition, several levels of multiresolution are possible, enabling us to identify object much larger than the query object. For objects smaller than the query, we just use a smaller tile size by reduction each dimension by two for each level of reduction.

Distance Measures	**Definitions**
Manhattan or l_1	$\sum \lvert x_i - y_i \rvert$
Euclidean or l_2	$\sum (x_i - y_i)^2$
Chebychev or l_∞	$\max \lvert x_i - y_i \rvert$
Kullback-Leibler Divergence	$\sum x_i \log(x_i/y_i)$
Chi-Square	$\sum \frac{x_i^2 - y_i^2}{y_i}$

Table 1: Table of different types of distance measures. $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ are the two feature vectors. The range of summation or maximum is always from 1 to n.

With the feature database in place, we seek out the feature vectors in the database that are "similar" to the feature vector corresponding to the query image. To properly define the notion of similarity, we assume that there is a distance, or dissimilarity, function associated with each type of feature. Two feature vectors that are a small distance apart are regarded to be more similar to each other than those with a large distance between them. We use some of the most common distance functions as described in table 1.

Inductive machine learning techniques can also be used to improve the retrieval of objects similar to a query. To do this, we first need to create a training set with positive and negative examples. One alternative we have tried is to present the top results of a distance-based similarity search and ask the user to assign positive and negative labels to the results based on the perceived similarity to the query. These labeled results are then used to train a classifier, which is applied to the entire feature database. The classifier assigns positive and negative labels to each feature vector in the database and the positive examples are presented to the user. This approach has produced improvements over

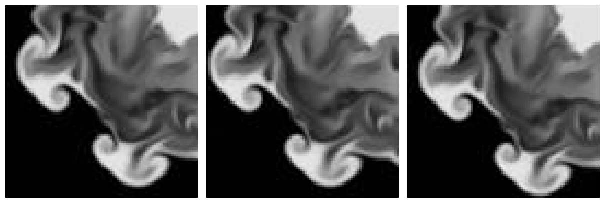

Figure 2: The results of applying the naive Bayes classifier to retrieve objects similar to a query. The first figure is the query. A hand-tuned distance-based search retrieved the query and the second figure. A naive Bayes classifier returned all three results. Note that the figures two and three are slightly displaced versions of the query.

simple distance-based similarity searches as illustrated in figure 2. The figure shows three objects that were retrieved by a naive Bayes trained using positive examples identified by a simple distance-based search and validated by the user. The similarity search was able to retrieve only the query and the first object shown in the feature. The trained naive Bayes retrieved two additional objects and did not return any false positives.

Having established that machine learning techniques can improve the retrieval accuracy in the context of our application, we next attempted to reduce the number of features being used in the similarity search. In our experiences we found that the performance of the system strongly depends on the choice of features and some experimentation is required to find a set of features that produce acceptable results. Moreover, the best choice of features for one query may not be appropriate for a different query. Thus, it seemed appropriate to investigate the role of feature selection in improving the retrieveal in the SBOR system. Fewer features would not only help address the "curse of dimensionality" problem, but also enable more efficient retrieval through the use of advanced data structures. In this paper, we compare and contrast the different feature selection methods we used in the context of similarity-based object retrieval in simulation data.

3 Methods

This section briefly describes the data and the algorithms used in this study. More detailed descriptions are available elsewhere [1].

3.1 Simulation Data. For the work in this paper, we consider the data from a high resolution 3-D shock tube simulation performed on a $2048 \times 2048 \times 1920$ grid over 27,000 time steps, obtained on 960 nodes of the IBM-SP Sustained Stewardship TeraOp system at Lawrence Livermore National Laboratory [7]. At the beginning of the simulation, two gases are separated by a membrane in a tube; then the membrane is pushed against a wire mesh. The simulation models the resulting mixing of the two gases.

Several variables are output by the simulation at each grid point at each time step. These variables include pressure, density, and velocity. In this work, we focus on the entropy, which is available as one byte of information per grid point. The entropy is scaled linearly with a minimum of 0 and a maximum of 255.

3.2 Features. The features included in the SBOR system range from simple pixel statistics to complicated visual attributes such as shape and texture. They provide a general and compact description of the distribution of pixel values inside a tile image. We focus primarily on features that are scale, rotation, and translation invariant. The features used in our experiments include:

Simple Features This set of features consists of the mean, the standard deviation, the maximum, and the minimum of all pixel values in a tile image.

Histogram This is a 16-bin histogram of pixel values in a tile image. The bins are uniform across the dynamic range.

ART The Angular Radial Transform (ART) belongs to a broad class of shape analysis tools based on moments [8]. ART projects a two-dimensional signal within the unit circle onto a set of complex orthonormal basis. Our implementation of ART is based on the region-shape descriptor defined in MPEG-7 [5, ch. 15]. functions. Following the MPEG-7 standard, twelve angular basis and three radial basis are used. This results in a 35-dimensional feature vector as the first normalized coefficient is always one.

Binary ART The Binary ART feature is just the ART feature applied to the tile in which the objects have been extracted by segmentation using simple thresholding.

Using all the features above results in a feature vector with 90 elements. In addition, some of the experiments include the following features:

Simple Geometry The simple geometry feature contains the parameters of the ellipse that best represents the distribution of the pixel values. It consists of five numbers: the location of the centroid, the lengths of the major and minor axes, and the angle between the major and the x axes. To make this feature scale invariant, the coordinates of the centroid are normalized by the dimension of the tile image, and the lengths of the major and minor axes are normalized by the diagonal of the tile image.

Binary Simple Geometry This feature is just the simple geometry feature applied to a tile in which the objects have been extracted by segmentation using simple thresholding. All object pixels are set equal to 255, and the background to 0.

Geometric Moments The geometric moment features are the seven functions derived from the normalized central moments of an image as defined by Hu in [2]. These moment funcitions are used to represent the shape of an image. They are scale, translation and rotation invariant, making them ideal for use in our application. As the values of these moment functions can be quitre different, we use the natural logarithm of the absolute value of the functions instead of the functions directly.

Binary Geometric Moments This is the Geometric moments feature applied to the tile in which the objects have been extracted by segementation using simple thresholding.

Using these additional features results in a feature vector with 114 elements.

3.3 Derived features. The feature vectors in the feature database contain fairly low-level features such as ART coefficients and the bin counts for the histograms. We first tried using these low-level features directly as input to machine learning techniques. However, we realized that the individual elements in the feature vectors do not represent anything meaningful in this context. Consider, for example, basing a discrimination on the 10-th bin of the histogram or on the fourth ART coefficient. So, we also computed a set of "derived" features which are different distances between features of the query and features of each element in the database. The distances we used are described in table 1. For example, a derived feature might be the L1 distance between the intensity histogram of the query and the histogram of each example in the feature database. There is a new derived feature vector for each feature vector in the database.

Calculating derived features also has the advantage of reducing the dimensionality of the problem. For example, the first data set used in the experiments has feature vectors with 90 elements, but only 20 derived features. The derived features corresponding to the same original feature as highly correlated. However, there is no obvious way to choose among the different distances. An exploratory data analysis showed that simple thresholding on any one of these distances would result in several false positive results, but several of these distances considered together would reduce the number of errors.

We do however note that using distances as the derived features implies that the features are dependent on the query, and must be recalculated as the query changes.

3.4 Data Preparation. For our experiments, we selected six representative 2-D slices of the simulation data. For testing purposes, we applied the following six transformations to each of the images:

- Anti-clockwise rotation by 36^o (rot36), 90^o (rot90), and 150^o (rot150).
- Reflection about the vertical axis in the middle (flip).
- Morphological erosion (erode) and dilation (dilate) by a 3×3 cross-shaped element.

These transformations allow us to mimic the behavior of the data as the simulation evolves. Examples of these transformation together with the original image are shown in Figure 3.

We follow the procedure described in Section 2 for testing: a 64×64 tile window is moved across each transformed slice in a step-size of eight pixels horizontally and vertically.

3.5 Training data. To compare systematically different feature selection algorithms, we follow a different approach to generate the training sets. Instead of a tedious and error-prone manual labelling, we identify as the "ground truth" all the feature vectors that correspond to tiles that overlap more than 90% with the query tile. A partial-overlap criteria is used because first, features should be robust against small translations induced by partial overlapping, and second, complete overlap is simply unachievable for some of the transformations such as rot36 and rot150. The positive examples for the training set come from the ground-truth examples. The negative examples are obtained by sampling randomly from the feature vectors that correspond to tiles with less than 50% overlap with the query tile. We ensure that the training and testing sets are disjoint. This procedure guarantees that the positive examples are visually similar with the query image, but we recognize that the procedure may omit relevant tiles that are not near the query tile.

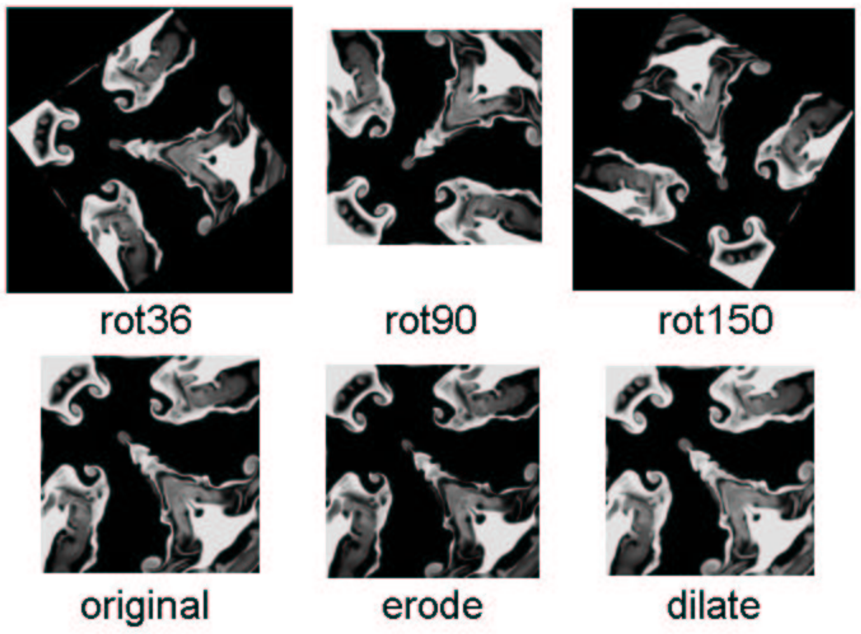

Figure 3: One of the original 2-D slices and the corresponding five spatial transformations.

3.6 Feature Selection Algorithms. One approach to feature selection is to preprocess the data and select features based on properties that good feature sets are presumed to have, such as orthogonality and high information content. This is known as the filter approach [4]. An alternative to preprocessing the data is the wrapper approach. The key idea is to consider the induction algorithm as a black box to be used by a search algorithm to evaluate each candidate feature subset [4]. The feature subset with the higher evaluation is selected as the final set on which to run the inducer. The resulting classifier should then be tested on data not used during the search.

Numerous search algorithms have been used to search for feature subsets [3]. Greedy algorithms that add or delete a single feature from the candidate feature subset are common. There are two basic variants: forward selection (FS) and backward elimination (BE). Forward selection starts with an empty set of features. It tentatively adds each feature that is not already in the candidate set and selects the feature that results in the highest estimated performance. The search finishes if no feature results in an improvement over the current subset. Backward elimination works in an analogous way, starting from the full set of features and tentatively deleting each feature not deleted previously.

Greedier versions of forward selection and backward elimination are possible. Instead of selecting or deleting the best of the features not already chosen, a greedier algorithm adds or deletes each candidate feature immediately if it produces an improvement over the current subset. These greedier versions explore fewer feature subsets, but risk getting trapped in local optima more easily than the standard algorithms. We decided to experiment with these greedier versions because we are interested in fast algorithms that can be used as part of an interactive system. These versions are sensitive to the order in which the features are considered to be added or deleted. We report the results of the best feature subset found in five random restarts that present the features to the algorithms in random order.

For our experiments, we chose a Naive Bayes classifier primarily because of its speed and simplicity. In our implementation, the naive Bayes assumes a normal distribution for the numeric features.

For each query, we present the training set to each feature selection algorithm. The wrappers use 10-fold crossvalidation to estimate the generalization of the naive Bayes with each proposed feature subset. The best feature subset found is used to train the naive Bayes using the entire training set. The trained classifier is then applied to the testing data that has not been seen by the algorithm until this point and different accuracy statistics are measured.

In addition to the wrapper algorithms we use principal component analysis (PCA). PCA produces mutually orthogonal linear combinations of the variables in the original data, such that the direction of the first principal component (PC) corresponds to the direction of maximum variance in the data, the direction of the second PC corresponds to the direction of second largest variance in the data, and so on. The data are usually standardized to have mean zero and variance one before computing the PCs to avoid the dominance of variables with large variances. The principal components are the eigenvectors of the data covariance matrix, with the first PC being the eigenvector corresponding to the largest eigenvalue. In many cases, the first few PCs explain most of the variability in the data and therefore provide a compact representation of the important features in the data. We chose to use enough PCs to explain 90% of the variability in the data.

We also adopted a method suggested by Mardia et al. [6] to use the PCs to eliminate unimportant variables. Starting with the eigenvector that corresponds to the smallest eigenvalue of the covariance matrix, we discarded the variable with the largest coefficient (in absolute value) in that vector. This variable is considered the least important. We then proceed to the eigenvector that corresponds to the second largest eigenvalue and discarded the variable with the largest coefficient, among the variables not discarded earlier. We continued with this process until we had identified the k most important variables, where k is the number of principal components that explained 90% of the variance of the original data.

All programs were developed in C++ and compiled with g++ version 3.1. The programs were executed on a single processor of a Linux (Red Hat 7.3) workstation with dual 2.4 GHz Intel Xeon processors and 512 Mb of memory.

4 Results

We performed two series of experiments. In the first series we used data at a single resolution, while in the second series we used multiresolution data.

All the experiments consider the six queries shown in figure 4. All the queries are 64×64 pixels.

4.1 Single Resolution Data. For the first series of experiments, we use the procedures described in the previous section to build training and testing sets for the original and modified images separately. Initially, we search for feature subsets and test the algorithms using the data from the original images exclusively. As explained in the previous section, the training and testing sets are disjoint, but in this experiment they are obtained from the same set of original images. The results of these experiments are in table 2. We observe that high precision and recall values can be obtained using all the features, but the results (especially the recall) improve by using the best feature subsets found by the search algorithms.

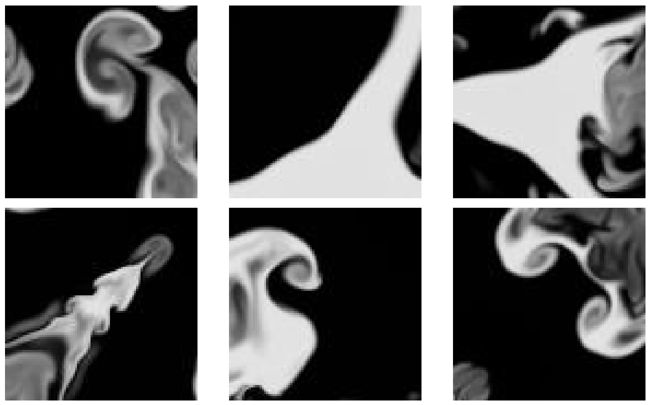
Figure 4: The six queries used in the experiments.

In the case of Query 6, the results are notably better, but there are few cases where using feature subsets result in slightly worse results.

In terms of the number of features selected, the results reported in table 3 show that forward selection clearly selects much fewer features than the other methods. As expected the backward elimination algorithms always selected many more features than the forward selection algorithms. The simple forward selection examined the fewer feature subsets, which means that it is the fastest of the algorithms that we considered.

A more challenging experiment is to use the data from the original images to search for feature subsets, train the Naive Bayes with the best subset of features from the original images, and test it on the modified images. The results are in table 4. As expected, the performance is worse than when the naive Bayes was tested on the original data. In particular, with queries 3 and 4 the previous experiment had perfect results, and in this experiment the naive Bayes could not identify a single instance as positive when using all the features. However, in most cases, the feature selection algorithms identified feature subsets that resulted in notable improvements.

The forward selection algorithm again delivers consistently good results over all the queries. The greedy forward selection sometimes delivers the best results, but in some cases (Q3 and Q4) it delivers much worse results than FS.

The size of the resulting feature subsets for this ex-

	Q1		Q2		Q3		Q4		Q5		Q6	
Method	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall
ALL	97.44	86.36	100.0	94.59	100.0	100.0	100.0	100.0	96.55	90.32	98.04	42.74
FS	97.44	97.44	97.14	94.44	100.0	100.0	100.0	100.0	96.55	93.33	94.12	69.57
BE	97.44	90.48	100.0	94.59	100.0	100.0	100.0	100.0	96.55	90.32	98.04	58.82
Greedy FS	97.44	79.17	94.29	94.29	100.0	100.0	100.0	100.0	96.55	91.80	94.12	60.00
Greedy BE	97.44	84.44	100.0	94.59	100.0	100.0	100.0	100.0	96.55	88.89	94.12	68.57
PCA	92.88	87.80	94.11	88.88	95.12	95.12	96.87	96.87	96.61	87.69	96.07	52.12
PCA Filter	100.0	82.97	94.28	91.66	100.0	97.61	100.0	100.0	98.28	87.69	94.11	50.00

Table 2: Results of experiments training and testing with data from the original images.

Method	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
ALL	20	20	20	20	20	20
FS	2	2	2	2	4	4
BE	19	19	18	19	18	13
Greedy FS	12	3	2	7	7	9
Greedy BE	18	19	18	17	18	6
PCA	4	5	4	5	4	5
PCA Filter	4	5	4	5	4	5

Table 3: Number of features selected on experiments training and testing with the 20 derived features from the original images.

periment are presented in table 5. The standard forward selection selected the smallest feature subset, while the backward elimination algorithms selected most of the features. PCA and PCA filter use only the first three or four PCs, but their accuracy on some of the queries (especially Q1–Q4) is inferior to the wrapper methods.

Since we intend to use feature selection as part of an interactive system, the execution times are important. On these experiments with single resolution data, PCA and the standard forward selection algorithm require less than one second to reach their solutions. The backward elimination algorithms require the longest times, and the greedy version (using five restarts) is the slowest algorithm, requiring approximately 15 seconds to run.

4.2 Multiresolution Data. For the second series of experiments we use the multiresolution data described in the previous section. These data include data from the original as well as from the modified images. We experimented with the derived features (distances) as we did in the previous set of experiments as well as with the original features extracted from the tiles.

The precision and recall results with the derived features are presented in table 6. We observe the same trends as with the previous experiments: Using all the features gives good results but using the selected feature subsets can improve the results. Again, forward selection gives most of the best results and always selects the smallest feature subset, as can be observed from table 7. The backward elimination algorithms keep most of the features.

In terms of execution time, the sequential feature selection is the fastest algorithm, requiring 1–2 seconds. PCA and PCA Filter require approximately the same time as the greedy forward selection and backward elimination (approx. 20–30 seconds). The multi-restart greedy BE is again the slowest algorithm (approx. 40 seconds).

Using the original raw features is attractive because it would eliminate the computation of derived features. While this computation is not too burdensome, eliminating it may result in an improvement in performance. The precision and recall obtained with the raw features are shown in table 8. These results suggest that the quality of the results remains approximately equal to using the derived features, when all the features are input to the naive Bayes. However, the performance after selecting a subset of features is generally better when derived features are used.

Once again, forward selection selected the fewest features and was the fastest algorithm. As expected because of the higher dimensionality, PCA is now the slowest algorithm, requiring approximately 65 seconds to run.

	Q1		**Q2**		**Q3**		**Q4**		**Q5**		**Q6**	
Method	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall
ALL	23.53	98.25	79.27	97.45	0.00	0.00	0.00	–	91.82	97.82	96.05	86.10
FS	72.69	98.86	77.20	98.03	95.28	99.51	61.93	97.83	96.68	98.44	91.19	92.59
BE	20.59	98.00	79.27	97.45	0.00	0.00	0.00	–	91.30	98.08	96.05	89.77
Greedy FS	75.63	96.77	94.82	97.86	19.81	100.0	41.74	98.91	92.33	98.90	94.53	89.88
Greedy BE	18.49	97.78	90.16	97.75	0.00	0.00	79.36	99.43	91.56	98.35	92.10	91.27
PCA	0	–	49.22	95.95	0	–	46.29	97.08	89.37	96.10	91.18	91.18
PCA Filter	0	–	38.86	98.68	0	–	46.33	99.01	90.79	98.06	93.61	88.25

Table 4: Results of experiments training with data from the original images and testing with data from the modified images.

Method	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
FS	2	2	2	3	4	7
BE	19	19	18	18	19	12
Greedy FS	5	2	2	4	6	8
Greedy BE	18	17	19	19	17	4
PCA	3	4	3	4	3	4
PCA Filter	3	4	3	4	3	4

Table 5: Number of features selected on experiments training with 20 derived features from the original data and testing with data from the modified images.

	Q1		**Q2**		**Q3**		**Q4**		**Q5**		**Q6**	
Method	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall
ALL	85.71	100.0	95.65	78.57	100.0	84.21	100.0	76.92	97.73	67.19	100.0	36.89
FS	92.86	100.0	95.65	91.67	100.0	100.0	100.0	100.0	95.45	91.30	52.63	54.05
BE	85.71	100.0	95.65	81.48	100.0	80.00	100.0	76.92	97.73	67.19	100.0	43.68
Greedy FS	78.57	100.0	95.65	81.48	100.0	76.19	100.0	100.0	97.73	91.49	60.53	76.67
Greedy BE	85.71	92.31	95.65	81.48	100.0	80.00	100.0	76.92	97.73	84.31	100.0	50.00
PCA	60.00	75.00	100.0	73.08	93.33	100.0	84.62	100.0	100.0	92.31	93.02	61.54
PCAFilter	100.0	70.00	100.0	45.45	100.0	86.96	84.62	91.67	97.50	86.67	100.0	35.45

Table 6: Results of experiments with multiresolution data using the 40 derived features.

Method	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
FS	3	2	2	2	1	4
BE	38	39	37	39	38	32
Greedy FS	4	9	4	2	4	5
Greedy BE	37	25	35	39	14	12
PCA	9	9	9	9	9	8
PCA Filter	9	9	9	9	9	8

Table 7: Number of feature subsets of experiments training and testing with multiresolution data using the 40 derived features.

	Q1		Q2		Q3		Q4		Q5		Q6	
All	85.71	75.00	100.0	71.43	100.0	80.00	84.62	84.62	97.50	75.00	100.0	59.09
FS	42.86	100.0	80.00	85.71	80.00	88.89	76.92	90.91	87.50	94.59	76.92	73.17
BE	85.71	75.00	100.0	71.43	100.0	80.00	84.62	84.62	97.50	78.00	100.0	59.09
Greedy FS	71.43	90.91	93.33	87.50	100.0	100.0	84.62	91.67	97.50	97.50	97.44	84.44
Greedy BE	85.71	75.00	100.0	71.43	100.0	80.00	84.62	84.62	97.50	81.25	100.0	66.10
PCA	94.44	94.44	100.0	100.0	100.0	100.0	100.0	92.31	100.0	86.67	97.37	63.79
PCAFilter	100.0	70.00	100.0	45.45	100.0	86.96	84.62	91.67	97.50	86.67	100.0	35.45

Table 8: Results of experiments with multiresolution data using the original 114 features.

Method	Q1	Q2	Q3	Q4	Q5	Q6
forward	3	3	3	3	6	7
backward	112	113	113	113	112	113
greedy forward	5	8	13	6	11	22
greedy backward	112	112	113	112	105	103
PCA	9	10	10	10	9	9
PCAFilter	9	10	10	10	9	9

Table 9: Number of feature subsets of experiments training and testing with multiresolution data using the original 114 features.

5 Summary and Conclusions

High-resolution simulations are useful to study complex phenomena, but produce large amounts of data that require specialized tools to be analyzed effectively. A common action in the analysis of simulation data is to find objects that are similar to an object that the human analyst finds intesting. For this task, we are building a similarity-based object retrieval system that uses (among other options) classification algorithms to retrieve objects similar to a query. This paper presents results of an experimental comparison of six methods that attempt to reduce the dimensionality of the input to the classifier.

We compared four wrapper feature selection algorithms to PCA and a variable elimination method based on PCA. We performed experiments with single- and multi-resolution data. The experimental results suggest that, in all cases, standard forward selection returns the smallest feature subsets in the shortest time and these feature subsets generally result in high recall and precision. For these reasons, forward selection will likely become the default option in our system.

We are currently investigating the performance of the feature selection algorithms using other classifiers that can be used in the SBOR system, such as decision trees and nearest neighbors. Future work will consider other dimensionality reduction methods such as random projections as well as incorporating active learning techniques to use the human analyst's time as effectively as possible.

Acknowledment

UCRL-CONF-200095. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[1] E. Cantú-Paz and C. Kamath. Retrieval of similar objects in simulation data using machine learning techniques. In *Proceedings of the Conference on Image Processing: Algorithms and Systems III.*

[2] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

[3] A. Jain and D. Zongker. Feature selection: evaluation, application and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

[4] G. John, R. Kohavi, and K. Phleger. Irrelevant features and the feature subset problem. In *Proceedings of the 11th International Conference on Machine Learning*, pages 121–129. Morgan Kaufmann, 1994.

[5] B.S. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7: Multimedia Content Description Interface.* John Wiley & Sons, Ltd., 2002.

[6] K.V. Mardia, J. T. Kent, and J.M. Bibby. *Multivariate Analysis.* Academic Press, 1995.

[7] A. Mirin et al. Very high resolution simulation of compressible turbulence on the IBM-SP system. Technical Report UCRL-JC-134237, Lawrence Livermore National Laboratory, 1999.

[8] R. Mukundan and K. R. Ramakrishnan. *Moment Functions In Image Analysis: Theory and Applications*. World Scientific, 1988.

[9] W. Rider et al. Using Richtmyer-Meshkov driven mixing experiments to impact the development of numerical methods for compressible hydrodynamics. In *Proceedings of the Ninth International Conference on Hyperbolic Problems Theory, Numerics, Applications*, pages 84 – 88, 2002. http://www.acm.caltech.edu/hyp2002/program.html.